Noise-Adaptive Ansatz Design

Variational Quantum Algorithms · Module 9 — Hardware Noise Models

Node 9.6

Position in Knowledge Graph

System: Variational Quantum Algorithms **Structural Domain:** Hardware Noise Models **Node:** 9.6

Noise mitigation (node 9.4-9.5) treats noise as a fact to correct *after* the circuit runs. **Noise-adaptive ansatz design** takes the opposite view: modify the ansatz *before* the circuit runs, so that the noise profile of the specific hardware has the least harmful effect.

The motivating observation: noise on real hardware is *heterogeneous*. Some qubits are better than others. Some two-qubit gates (on specific qubit pairs) have much higher fidelity than others. The hardware connectivity graph has topology (not every pair of qubits can interact directly). And the noise spectrum changes slowly over time with drift and recalibration.

A noise-blind ansatz ignores all this and treats the circuit as a uniform object. A noise-adaptive ansatz uses the hardware information to:

1. **Place the best-fidelity qubits where the circuit needs them most.**
2. **Route two-qubit gates through the lowest-noise pairs.**
3. **Reduce depth in regions where coherence is weakest.**
4. **Choose the ansatz shape itself** to match the hardware's native gate set.

The result: better circuit fidelity for the same logical operation, at zero quantum runtime cost (all the optimization is classical). This is essentially a form of *hardware-aware compilation* specialized for VQAs.

This node covers the main techniques: qubit selection, routing, Hamiltonian-to-hardware mapping, hardware-efficient ansätze, and the ADAPT-VQE family of noise-aware ansatz growth.

Qubit Selection

The simplest form of noise-adaptive design. Modern superconducting hardware has 50-1000 qubits; most VQA problems use far fewer. The question: which subset of qubits do you pick?

Naive approach: use qubits 0 through $n-1$ on the device. Simple but ignores fidelity differences.

Better approach: rank qubits by fidelity metrics and pick the best n.

Useful metrics for ranking:

- **T1, T2 coherence times** — higher is better.
- **Readout fidelity** — higher is better.
- **Single-qubit gate fidelity** — higher is better.
- **Two-qubit gate fidelity on connecting edges** — higher is better.
- **Calibration freshness** — more recently calibrated qubits are usually better.

The ranking is done per-VQA-run using the daily calibration data published by the hardware provider (IBM, Rigetti, IonQ, etc.). The best qubits change over time as the hardware drifts.

Gotcha: the "best" qubits individually may not have the best two-qubit couplings. You need to optimize over *subgraphs*, not individual qubits. This is a graph optimization problem and can be hard in general (subgraph isomorphism is NP-hard), but practical heuristics work well for small problem sizes.

Reference. Tannu-Qureshi (2019) showed 3-5x fidelity improvement on IBM hardware via hardware-aware qubit selection alone.

Gate Routing

A VQA circuit usually implies a *logical* connectivity pattern (which qubits interact in the cost Hamiltonian). The hardware has a *physical* connectivity graph (which qubits are directly coupled). These rarely match exactly.

The **routing problem** is: map logical qubits to physical qubits, inserting SWAP gates where needed, to realize the logical circuit on the physical hardware with minimum noise cost.

Naive routing minimizes the total SWAP count — fewer SWAPs means fewer noisy operations.

Noise-aware routing weights each potential SWAP by the error rate of the specific two-qubit gate used. A high-fidelity SWAP pathway is preferred even if it's longer.

Formalization. This is a weighted graph optimization problem. Given the logical connectivity graph G_L and the physical graph G_P with edge weights w_{ij} (error rates), find a mapping and SWAP sequence that minimizes the total weighted error.

Practical tools. Qiskit's `SabreSwap`, `SabreLayout`, and `NoiseAdaptiveLayout` passes do noise-aware routing. Similar tools exist in Cirq (Google) and tket (Quantinuum).

Impact. On hardware with significant fidelity heterogeneity, noise-aware routing gives 2-5x fidelity improvement compared to naive routing.

Hardware-Efficient Ansätze (HEA)

A deeper form of adaptation: choose the ansatz *shape* to match the hardware's native gate set and connectivity.

Standard HEA:

$$U(\boldsymbol{\theta}) = \prod_{l=1}^{L} \left[\left(\prod_i R_Y(\theta_i^l) R_Z(\phi_i^l) \right) \cdot \left(\prod_{(i,j)\in E} \mathrm{CNOT}_{ij} \right) \right],$$

where E is the hardware's native two-qubit connectivity. Key properties:

- **Uses only native gates** — no expensive decomposition of arbitrary U into single-qubit rotations and CNOTs.
- **Respects connectivity** — no SWAP insertion required, since two-qubit gates are only between connected qubits.
- **Linearly parameterized in layers** — depth is the main knob.

Tradeoff. HEA has the best hardware fidelity *but* the worst expressibility per parameter. Hardware-native gates often don't span the full unitary group cleanly, so you need more layers to reach the same expressibility as a problem-tailored ansatz.

Practical observation. For NISQ VQAs, the fidelity savings often outweigh the expressibility loss. HEA with $L = 5 - 20$ layers is competitive with much deeper "theoretically better" ansätze on noisy hardware.

Reference. Kandala et al. (2017) — the original hardware-efficient VQE on IBM hardware for small molecules.

Hamiltonian-to-Hardware Matching

A more sophisticated technique: match the *structure* of the Hamiltonian to the hardware topology.

Observation. A chemistry Hamiltonian has locality structure — some terms involve adjacent orbitals, others involve distant ones. Mapping this to the hardware:

1. **Place orbitals that interact strongly on well-connected physical qubits.**
2. **Use SWAP networks that match the locality pattern.**
3. **Schedule the simulation to minimize communication between distant qubits.**

For Hamiltonians derived from lattice models (Hubbard, Heisenberg, Ising), this is even cleaner: the lattice structure directly maps to a hardware connectivity pattern, and the best ansatz is one that mirrors that lattice.

For chemistry VQEs specifically, the **unitary coupled cluster (UCC)** ansatz has a natural locality pattern inherited from Slater-Condon rules. Recent work (Lee et al. 2021) shows that matching UCC excitation operators to hardware topology improves fidelity 2-10x for 10-20 qubit molecules.

For QAOA on graph problems, a good rule is to *assign graph vertices to physical qubits such that adjacent vertices are adjacent on the hardware*. This reduces the SWAP count dramatically on native problems.

ADAPT-VQE And Growing Ansätze

The **Adaptive Derivative-Assembled Pseudo-Trotter (ADAPT-VQE)** method grows the ansatz one gate at a time, selecting at each step the gate that gives the largest gradient component.

Procedure.

1. Start with an empty ansatz (just the reference state).
2. Compute the gradient of the cost with respect to adding each candidate gate from a pre-defined pool.
3. Add the gate with the largest gradient magnitude.
4. Optimize the parameters of the current ansatz.
5. Repeat until convergence.

Why this is noise-adaptive. The gradient estimate is based on the noisy circuit, so the *noise-favored* gates tend to be selected — gates that are high-fidelity on the specific hardware give larger and more reliable gradients, so they enter the ansatz first. The result is an ansatz that is implicitly hardware-aware.

Pros: extremely compact ansätze (much fewer parameters than HEA), naturally hardware-aware, grows only as much as needed for the specific problem.

Cons: the gradient computation at each growth step is expensive (you're computing the gradient over the full operator pool). For large pools, this becomes a significant overhead.

Reference. Grimsley, Economou, Barnes, Mayhall (2019) introduced ADAPT-VQE. Subsequent work extended it to fermionic-qubit mappings and noise-aware variants.

Gate Fidelity As A Cost Term

A direct approach to noise-adaptive design: add the gate-noise cost directly to the training objective.

Idea. Define a modified cost:

$$C_{\text{adapted}} = C_{\text{quantum}} + \alpha \cdot C_{\text{noise}},$$

where $C_{\text{noise}} = \sum_g p_g$ is the sum of noise rates over all gates in the circuit, and α is a regularization parameter.

Optimization now trades off the quantum cost against the noise cost: the optimizer is encouraged to find low-gate-count, low-error solutions.

Caveat. The gate count is discrete (number of gates in the circuit), so gradient descent doesn't directly apply. You need either: - A soft penalty (α times a smooth approximation of gate count). - Discrete search (genetic algorithms, reinforcement learning, Bayesian optimization). - A structured ansatz family where gate count is controlled by a continuous parameter.

Practical use. Primarily a research direction — not yet standard. The thesis's HOPSO framework can naturally incorporate a noise-aware cost term in its objective generator, which is a potential future extension.

Dynamic Ansatz Reconfiguration

A futuristic version: the ansatz structure is **reconfigured during training** based on observed hardware fidelity changes.

Procedure.

1. Monitor the hardware fidelity in real time (via periodic benchmarking).
2. Detect when specific gates have degraded (e.g., due to drift or a recalibration).
3. Re-route or re-layer the ansatz to avoid the degraded gates.
4. Continue optimization with the new ansatz structure.

This is a form of **adaptive circuit compilation during training** and fits naturally into the control-theoretic view of Module 7. It's the logical extension of node 7.6's feedback control framework to the ansatz level.

Research status. Early experiments (Ziman et al. 2023, Hetzinger et al. 2024) show this can improve final fidelity by 2-5x on drifting hardware. Not yet standard; active research.

Noise-Adaptive Vs Mitigation: Structural Comparison

Dimension	Mitigation (9.4, 9.5)	Adaptation (9.6)
When applied	After runs	Before runs
Cost domain	Shot count	Classical compile time
What it fixes	Residual bias	Systematic fidelity
Depth limit	Extends by 3-10x	Extends by 2-5x
Combines with	Other mitigation	Mitigation techniques
Hardware info needed	Minimal	Full calibration data
Best for	Moderate noise	Heterogeneous noise

The two approaches are complementary. You should apply *both*: a noise-adaptive ansatz to minimize the underlying error, plus mitigation to further reduce residual bias. The combined improvement is roughly multiplicative.

The Thesis View

Noise-adaptive design fits the thesis framework in a clean way:

1. **It is a structural modification of the objective generator.** The objective function C is unchanged conceptually, but its hardware representation changes.
2. **HOPSO can incorporate hardware data as a parameter.** The optimizer can be given the current fidelity map as a static input, and it can be designed to prefer parameter settings that activate high-fidelity gates more than low-fidelity ones.
3. **Separation of dynamics and objective.** The *dynamics* of HOPSO (how particles move) is unchanged; only the *objective landscape* (what they're searching) is adapted. This is the right way to think about it architecturally.
4. **Adaptive regime.** The static noise-adaptive design of section 9.6 extends naturally to the dynamic one of Module 10 — the objective generator can be updated during training as hardware drifts, closing the loop between hardware monitoring and circuit structure.

Structural Role

This node is the **ansatz-level mitigation** complement to nodes 9.4-9.5 (post-processing mitigation). It covers qubit selection, gate routing, hardware-efficient ansätze, Hamiltonian-to-hardware matching, ADAPT-VQE, and dynamic reconfiguration.

It is the structural precondition for Module 10's adaptive VQA architectures, which take the noise-adaptive ideas and embed them in a feedback-control framework.

Relations to Other Concepts

- **Tannu, Qureshi (2019)** — hardware-aware qubit selection.
- **Kandala et al. (2017)** — original hardware-efficient ansatz.
- **Grimsley et al. (2019)** — ADAPT-VQE.
- **Li, Benjamin, Proctor (2020)** — noise-aware compilation with SabreSwap-style algorithms.
- **Module 9.4 (Mitigation)** — complementary post-processing approach.
- **Module 7.6 (Feedback control)** — dynamic reconfiguration as closed-loop control.
- **Module 10 (Adaptive VQA)** — the deployment endpoint.

Worked Example — 8-Qubit Chemistry VQE On A Heterogeneous Device

Setup. An 8-qubit H_2O chemistry VQE on a 27-qubit IBM device (call it `ibm_kyoto` for concreteness). Hardware calibration data: - Single-qubit gate fidelities: 99.95%-99.99%. - Two-qubit gate fidelities: 98.5%-99.6% (varies by edge). - Best 8-qubit connected subgraph: qubits $\{3, 5, 8, 11, 14, 16, 19, 22\}$ with average two-qubit fidelity 99.4%. - Worst 8-qubit connected subgraph: $\{0, 1, 2, 4, 6, 7, 9, 10\}$ with average two-qubit fidelity 98.8%.

Circuit. A UCCSD ansatz with 50 two-qubit gates. Run with 10^5 shots per cost evaluation.

Comparison:

Strategy	Qubit subset	Avg 2q fid	Circuit fid	Cost bias
Naive (qubits 0-7)	worst	0.988	0.55	45%
Noise-aware selection	best	0.994	0.74	26%

Strategy	Qubit subset	Avg 2q fid	Circuit fid	Cost bias
Best + noise-aware routing	best	0.994 + routing	0.78	22%
Best + routing + HEA switch	best	—	0.83	17%
Best + routing + HEA + ZNE	best	—	—	6%
Best + routing + HEA + ZNE + CDR	best	—	—	2%

Observation. Noise-adaptive design alone reduces bias by ~30% (45% → 17%). Adding mitigation on top reduces it further (17% → 2%). The stack multiplies: the total improvement is ~20x.

Compute cost. Qubit selection and routing: seconds. HEA conversion: minutes. Training: same as baseline. ZNE: 3x shots. CDR: 1.5x more compute for Clifford regression. Net: ~5x compute cost total for 20x bias reduction. Worth it.

Visualization

Pending. A diagram of a 27-qubit device with two 8-qubit subsets highlighted: the naive one (bottom-left, mostly-worst) and the noise-aware one (best-fidelity subset). Edges are colored by two-qubit gate fidelity. Caption: "Qubit selection: picking the right 8 out of 27 matters."

Exercises

1. Given a hardware graph $G_P = (V, E, w)$ with edge weights w_{ij} (error rates) and a desired subset size n, formulate the noise-aware qubit selection problem as a graph optimization problem. Is it tractable?
2. For a QAOA on a 10-vertex ring-graph MaxCut, design the noise-aware qubit-to-vertex mapping on a linear-connectivity hardware device. Compare SWAP count to naive mapping.
3. (VQA) Implement a noise-aware variant of the parameter-shift gradient that weights each gate's gradient by its fidelity. Does this improve convergence on a noisy 6-qubit VQA?

Research Extensions

- **Adaptive noise-aware compilation** — recompile during training as hardware drifts.
- **Noise-aware parameter initialization** — start near parameter settings that favor high-fidelity gates.
- **Multi-objective ansatz search** — jointly optimize expressivity and hardware fidelity.
- **Hardware-co-designed ansätze** — design the ansatz *and* the hardware gate set together for specific target problems.
- **Transfer learning for ansatz structure** — reuse ansatz designs across similar problems on similar hardware.

Cross-links to Other Courses

- **Compilers and code generation** — routing and layout as compilation problems.
- **Graph theory** — subgraph selection and minimum-weight routing.
- **Machine learning architecture search** — ADAPT-VQE as a form of NAS (neural architecture search) for quantum circuits.
- **Module 9.4 (Mitigation)** — the post-processing complement.
- **Module 10 (Adaptive Control Systems)** — the closed-loop ansatz adaptation.

Concepts: quantum-noise, parameterized-circuits, graph-theory, adaptive-systems
Prerequisites: 9.2, 9.3, 9.4
Enables: 9.8